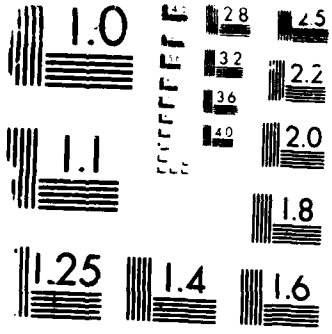
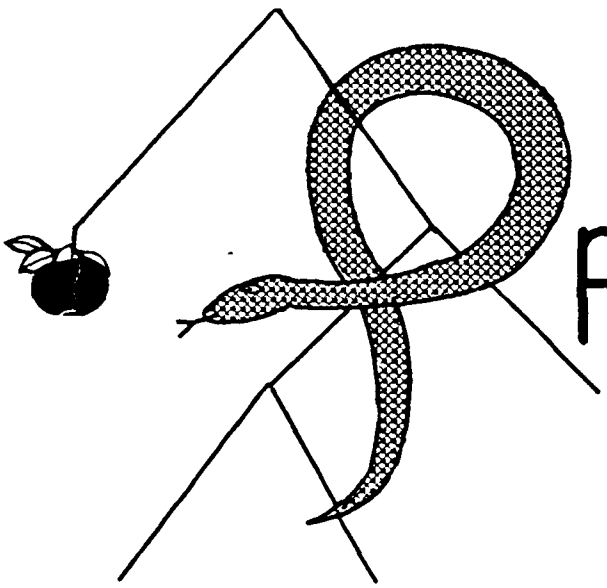
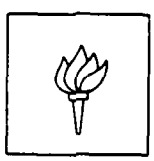
AD-A188 638 RESPONDING TO SEMANTICALLY ILL-FORMED INPUT(U) NEW YORK 1/1
UNIV NY COURANT INST OF MATHEMATICAL SCIENCES
R GRISHMAN ET AL SEP 87 PROTEUS-M-7-A N00014-85-K-0163
UNCLASSIFIED F/G 5/7 NL



RESOLUTION TEST CHART



PROTEUS PROJECT



Department of Computer Science
Courant Institute of Mathematical Sciences
New York University

Responding to Semantically Ill-formed Input

Ralph Grishman and Ping Peng
Computer Science Department
New York University
251 Mercer Street
New York, NY 10012

ABSTRACT

One cause of failure in natural language interfaces is semantic overshoot; this is reflected in input sentences which do not correspond to any semantic pattern in the system. We describe a system which provides helpful feedback in such cases by identifying the "semantically closest" inputs which the system would be able to understand.

1. Introduction

Natural language interfaces have achieved a limited success in small, well circumscribed domains, such as query systems for simple data bases. One task in constructing such an interface is identifying the relationships which exist in a domain, and the possible linguistic expressions of these relationships. As we set our sights on more complex domains, it will become much harder to develop a complete or nearly complete catalog of the relevant relationships and linguistic expressions; substantial gaps will be inevitable. In consequence, many inputs will be rejected because they fail to match the semantic/linguistic model we have constructed for the domain.

We are concerned with the following question: what response should we give a user when his input cannot be analyzed for the reasons just described? The response "please rephrase" gives the user no clue as to *how* to rephrase. This leads to the well-known "stonewalling" phenomenon, where a user tries repeatedly, without success, to rephrase his request in a form the system will understand. This may seem amusing to the outside observer, but it can be terribly frustrating to the user, and to the system designer watching his system being used.

We propose instead to provide the user with sentences which are semantically close to the original input (in a sense to be defined below) and are acceptable inputs to the system. Such feedback may occasionally be confusing, but we expect that more often it will be helpful in showing the system's capabilities and suggesting possible rephrasings.

In the remainder of this abstract we briefly review the prior work on responding to ill-formedness, describe our proposal and its implementation as part of a small question-answering system, and relate our initial experiences with this system.

2. Background

2.1. Relative and Absolute Ill-formedness

Weischedel and Sondheimer [Weischedel 1983] have distinguished two types of ill-formedness: *absolute* ill-formedness and *relative* ill-formedness. Roughly speaking, an absolutely ill-formed input is one which does not conform to the syntactic and semantic constraints of the natural language or the sublanguage; a relatively ill-formed input is one which is outside the coverage of a particular natural language interface. Our concern is primarily with relative ill-formedness. For complex domains, we believe that it will be difficult to create complete semantic models, and therefore that relatively ill-formed input will be a serious problem -- a problem that it will be hard for users to remedy without suitable feedback.

2.2. Syntactic and Semantic Ill-formedness

Earlier studies have examined both syntactically and semantically ill-formed input. Among the work on syntactically ill-formed input has been EPISTLE [Miller 1981], the work of Weischedel and Sondheimer [Weischedel 1981, Kwasney 1981, and Weischedel 1983], and Carbonell and Hayes [Carbonell 1983]. Some of this work has involved the relaxation of syntactic constraints; other (such as Carbonell and Hayes) a reliance primarily on semantic structures when syntactic analysis fails. Our system has been primarily motivated by our concern about the possibility of constructing complete semantic models, so we have focussed to date on semantic ill-formedness, but we believe that our system will have to be extended in the future to handle syntactic ill-formedness as well.

2.3. Error Identification and Correction

For some applications, it is sufficient that the point of ill-formedness be identified, and the constraint be relaxed so that an analysis can be obtained. This was the case in Wilks' early work on "Preference Semantics" [Wilks 1975], which was used for machine translation applications. In other applications it is necessary to obtain an analysis conforming to the system's semantic model in order for further processing of the input to take place, in effect "correcting" the user's input. This is the case for data base query (our current application), for command systems (such as MURPHY [Selfridge 1986]), and for message entry systems (such as NOMAD [Granger 1983] and VOX [Meyers 1985]).

2.4. System Organization

Error correction can be provided either by making pervasive changes to a set of rules, or by providing uniform correction procedures which work with a standard (non-correcting) set of rules. In the syntactic domain, EPISTLE is an example of the former, the metarule approach [Weischedel 1983] an example of the latter. We feel that, particularly for semantic correction, it is important to take the "uniform procedure" approach, since a semantic model for a large domain will be difficult enough to build and maintain without having to take the needs of a correction mechanism into account. It is equally important to have a procedure which will operate on a system with separate syntactic and semantic components, so that we may reap the advantages of such an organization (conciseness, modularity). The NOMAD system used procedures associated with individual words and so was very hard to extend [Granger 1983, p. 195]; the VOX system remedied some of these defects but used a "conceptual grammar" mixing syntactic and semantic constraints [Meyers 1985]. The MURPHY system [Selfridge 1986] is most similar to our own work in terms of the approach to semantic constraint relaxation and user feedback; however, it used a syntactic representation which would be difficult to extend, and required weights in the semantic model for the correction procedure.

In addition to the shortcomings of the systems just described, we felt it important to develop and test a system in order to gain experience in the effectiveness of these correction techniques. Although (as just noted) many techniques have been described, the published reports contain

virtually no evaluation of the different approaches.

3. System Overview

Our feedback mechanism is being evaluated in the context of a small question-answering system with a relatively standard structure. Processing of a question begins with two stages of syntax analysis: parsing, using an augmented context-free grammar, and syntactic regularization, which converts the various types of clauses (active and passive; interrogative, imperative, and declarative; relative and reduced relative; etc.) into a canonical form. In this canonical form, each clause is represented as a list consisting of: tense, aspect, and voice markers; the verb (root form); and a list of operands, each marked by "subject", "object", or the governing preposition. For example, "John received an A in calculus." would be translated to

(past receive (subject John) (object A) (in calculus))

Processing continues with semantic analysis, which translates the regularized parse into an extended-predicate-calculus formula. One aspect of this translation is the determination of quantifier scope. Another aspect is the mapping of each verb and its operands (subject, objects, and modifiers) into a predicate argument structure. The predicate calculus formula is then interpreted as a data base retrieval command. Finally, the retrieved data is formatted for the user.

The translation from verb plus operands to predicate plus arguments is controlled by the *model* for the domain. The domain vocabulary is organized into a set of verb, noun, adjective, and adverb semantic classes. The model is a set of patterns stated in terms of these semantic classes. Each pattern represents one combination of verb and operands which is valid (meaningful) in this domain. For example, the pattern which would match the sentence given just above is

(v-receive (subject nstudent) (object ngrade) (in ncourse))

where *v-receive* is the class of verbs including receive, get, etc.; *nstudent* the class of students; *ngrade* the class of grades; and *ncourse* the class of course names. Associated with each pattern is a rule for creating the corresponding predicate-argument structure.

4. The Diagnostic Process

In terms of the system just described, the analysis failures we are concerned with correspond to the presence in the input of clauses which do not match any pattern in the model. The essence of our approach is quite simple: find the patterns in the model which come closest to matching the input clause, and create sentences using these patterns. Implementation of this basic idea, however, has required the development of several processing steps, which we now describe.

Our first task is to identify the clauses to which we should apply our diagnostic procedure. Our first impulse might be to trigger the procedure as soon as we parse a clause which doesn't match the model. However, the process of matching clause against model serves in our system to check selectional constraints. These constraints are needed to filter out, from syntactically valid analyses, those which are semantically ill-formed. In a typical query we may have several semantically ill-formed analyses (along with one well-formed one), and thus several occasions of failure in the matching process before we obtain the correct analysis.

We must therefore wait until syntax analysis is complete and see if there is any syntactic analysis satisfying all selectional constraints. If there is no such analysis, we look for an analysis in which all but one clause satisfies the selectional constraints; if there is such an analysis, we mark the offending clause as our candidate for diagnostic processing.

Next we look for patterns in the model which "roughly match" this clause. As we explained above, the regularized clause contains a verb and a set of syntactic cases with case labels and fillers; each model pattern specifies a verb class and a set of cases, with each case slot specifying a label and the semantic class of its filler. We define a distance measure between a clause and a

pattern by assigning a score to each type of mismatch (clause and pattern have the same syntactic case with different semantic classes; clause and pattern include the same semantic class but in different cases; clause has case not present in pattern; etc.) and adding the scores. We then select the pattern or patterns which, according to this distance measure, are closest to the offending clause.

We now must take each of these patterns and build from it a sentence or phrase the user can understand. Each pattern is in effect a syntactic case frame, with slots whose values have to be filled in. If the case corresponds to one present in the clause, we copy the value from the clause; if the case is optional, we delete it. Othewise we create a slot filler consisting of an indefinite article and a noun describing the semantic class allowed in that slot (for example, if the pattern allows members of the class of students in a slot, we would generate the filler "a student"). When all the slots have been filled, we have a structure comparable to the regularized clause structure produced by syntactic analysis.

Finally each filled-in pattern must be transformed to a syntactic form parallel to that of the original offending clause. (If we don't do this -- if, for example, the input is a yes-no question and the feedback is a declarative sentence -- the system output can be quite confusing.) We isolate the tense, voice, aspect, and other syntactic features of the original clause (this is part of the syntactic regularization process) and transfer these features to the generated structure. If the offending clause is an embedded clause in the original sentence, we save the context of the offending clause (the matrix sentence) and insert the "corrected" clause into this context. We take the resulting structure and apply a sentence generation procedure. The generation procedure, guided by the syntactic features markers, applies "forward" transformations which eventually generate a sentence string. These sentences are presented as the system's suggestions to the user.

5. Examples

The system has been implemented as described above, and has been tested as part of a question-answering system for a small "student transcript" data base. The syntactic model currently has patterns for 30 combinations of verbs and arguments. While the model has been gradually growing, it still has sufficient "gaps" to give adequate opportunity for applying the diagnostics.

A few examples will serve clarify the operation of the system. The system has models

(take (subject student) (object course))

and

(offer (subject school) (object course))

but no model of the form

(offer (subject student) (object course))

Accordingly, if a user types

Did any students offer V11?

(where V11 is the name of a course), the system will respond

Sorry, I don't understand the pattern

(students offer courses)

and will offer the "suggestions"

Did any students take V11?

and

Did some school offer V11?

Prepositional phrase modfiers are analyzed by inserting a "be" and treating the result as a relative clause. For example, "students in V11" would be expanded to "students [such that] [students] be in V11". If the resulting clause is not in the semantic model, the usual correction procedures are applied. As part of our policy of limiting the model for testing purposes, we did not include a pattern of the form

(be (subject student) (in course))
but there is a pattern of the form
(enroll (subject student) (in course))
(for sentences such as "Tom enrolled in V11."). Therefore if the user types
List the students in V11.
the system will generate the suggestions
List the students who enroll in V11.
and
List the students.
(the second suggestion arising by deleting the modifier).

6. Current Status

The system has been operational since the summer of 1986. Since that time we have been regularly testing the system on various volunteers and revising the system to improve its design and feedback. We instructed the volunteers to try to use the system to get various pieces of information, rather than setting them a fixed task, so the queries tried have varied widely among users.

The experimental results indicate both the strength and weakness of the technique we have described. On the one hand, semantic pattern mismatch is not the primary cause of failure; vocabulary overshoot (using words not in the dictionary) is much more common. In a series of tests involving 375 queries (by 8 users), 199 (53%) were successful, 95 (25%) failed due to missing vocabulary, 22 (6%) failed due to semantic pattern mismatch, and 59 (16%) failed for other reasons. On the other hand, in cases of semantic pattern mismatch, the suggestions made by the system usually include an appropriate rephrasing of the query (as well as some extraneous suggestions). Of the 22 failures due to semantic pattern mismatch (in both series of tests), we judge that in 14 cases the suggestions included an appropriate rephrasing.

7. Assessment

These results, while not definitive, suggest that the technique described above *is* a useful one, but will have to be combined with other techniques to forge a general strategy for dealing with problems encountered in interpreting the input. In particular, we will have to extend our technique to deal with input containing unknown words. It should be possible to do this in a straightforward way by adding dictionary entries for the closed syntactic classes, guessing from morphological clues the syntactic class(es) of new words not in the dictionary, obtaining a parse, and then applying the techniques just described (with a new word treated as a semantic unknown, not belonging to any class).

Our system only offers suggestions; it does not aspire to *correct* the user's input. That would be an unreasonable expectation for our simple system, which does not maintain any user or discourse model. Our current system typically generates several equally-rated suggestions for an ill-formed input. For a more sophisticated system which does maintain a richer model, correction may be a feasible goal. Specifically, we might generate the suggested questions as we do now and then see if any question corresponds to a plausible goal.

8. References

[Carbonell 1983]

J. G. Carbonell and P. J. Hayes, Recovery Strategies for Parsing Extragrammatical Language. *Am. J. Computational Linguistics* **9** (3-4), 1983, pp. 123-146.

[Granger 1983]

R. H. Granger, The NOMAD System: Expectation-Based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-Formed Text. *Am. J. Computational Linguistics* **9** (3-4), 1983, pp. 188-196.

[Kwasney 1981]

S. C. Kwasney and N. K. Sondheimer, Relaxation Techniques for Parsing Ill-Formed Input. *Am. J. Computational Linguistics* **7**, 1981, pp. 99-108.

[Meyers 1985]

A. Meyers, VOX -- An Extensible Natural Language Processor. *Proc. IJCAI-85*, Los Angeles, CA, pp. 821-825.

[Miller 1981]

L. A. Miller, G. E. Heidorn, and K. Jensen, Text-critiquing with the EPISTLE System: An Author's Aid to Better Syntax. In *Proc. Nat'l Comp. Conf.*, AFIPS Press, Arlington, VA, 1981, pp. 649-655.

[Selfridge 1986]

M. Selfridge, Integrated Processing Produces Robust Understanding, *Computational Linguistics* **12** (2), 1986, pp. 89-106.

[Weischedel 1980]

R. M. Weischedel and J. E. Black, Responding Intelligently to Unparsable Inputs. *Am. J. Computational Linguistics* **6** (2), 1980, pp. 97-109.

[Weischedel 1983]

R. M. Weischedel and N. K. Sondheimer, Meta-rules as a Basis for Processing Ill-Formed Input. *Am. J. Computational Linguistics* **9** (3-4), 1983, pp. 161-177.

[Wilks 1975]

Y. Wilks, An Intelligent Analyser and Understander of English. *Comm. ACM* **18**, 1975, pp. 264-274.

END
DATE
FILMD
3-88
DTIC